|  |  |
| --- | --- |
|  | **Instituto Tecnológico de Costa Rica**  Ingeniería en Computadores/Bases de Datos (CE3101)  Primer Semestre 2019/*Proyecto #2* |

1. **Objetivo General**

Desarrollar una aplicación que permita manejar la descripción del caso expuesto en el punto 4.

1. **Objetivos Específicos**
   1. Aplicar los conceptos del modelo conceptual y relacional.
   2. Crear una Base de Datos en PostgreSQL.
   3. Crear un WebService/REST Service.
   4. Crear una página Web.
   5. Usar herramientas como PostgreSQL, Angular, Bootstrap, HTML5, CSS, y reporting services o cristal reports.
   6. Crear un plan de instalación.
   7. Crear un plan de proyecto.
2. **Datos Generales**
   1. El valor del proyecto: 20%
   2. Nombre código: TABAS
   3. La tarea debe ser implementada grupos de 4 personas.
   4. La **fecha de entrega:**
      1. Plan de trabajo: 14/Mayo/2019
      2. Resumen Ejecutivo Avance 1: 21/Mayo/2019
      3. Resumen Ejecutivo Avance 2: 28/Mayo/2019
      4. Resumen Ejecutivo Avance 3: 04/Junio/2019
      5. Funcionalidad completa: 08/Jun/2019.
   5. Cualquier indicio de copia será calificado con una nota de 0 y será procesado de acuerdo al reglamento.
3. **Descripción del caso de estudio**

Con el inesperado crecimiento de la compañía TecAirlines se ha vuelto caótico el manejo del equipaje de los estudiantes y pasajeros por lo que se hace completamente necesario un sistema de administración de equipaje (Tec Airlines Baggage Administration System).

Este sistema administrará todo lo relacionado a equipaje incluso la comunicación con el ministerio de hacienda de Costa Rica y la emisión de facturas electrónicas para equipaje. Es importante mencionar: en caso de retraso de la entrega de la aplicación la empresa contratada deberá pagar la multa impuesta por el gobierno costarricense de $20.000 mensuales por cada semana de atraso en la facturación digital.

* 1. **Requerimientos del Software**

**Debe ser desarrollada utilizando Angular, Bootstrap, CSS, HTML5, NO se permite utilizar el Entity Framework.**

* 1. **Requerimientos funcionales del Sistema**

**Aplicación Web Administración:**

* Inicio de Sesión: La aplicación estrictamente debe saber quien esta registrado por lo que se debe permitir el inicio de sesión.
* Registro de trabajadores: El sistema debe permitir el registro de trabajadores del que se almacenará el nombre, apellidos, numero de cedula y rol que desempeña (Embarcador, Scan, Administrador, Recepcionista -deben crear al menos 5 trabajadores por rol-).
* *Registrar Usuario:* el sistema debe permitir la creación de distintos usuarios del sistema. Esta funcionalidad será utilizada por los funcionarios de la aerolínea (Recepcionista) donde se solicitará información de contacto (Nombre completo, teléfono, correo y si es estudiante en caso de serlo solicitara la información de la Universidad donde estudia y su carnet que se utilizara para almacenar millas en su programa de lealtad).
* *Creación de Maletas:* El sistema debe poder crear maletas de las cuales se registrará: el Usuario, color, peso, costo de enviar esa maleta y super importante número de maleta. Por cada maleta generada en el sistema se generarán 2 archivos un XML y un PDF con la factura respectiva. El XML se deberá ‘enviar’ a Hacienda (el XML generado debe seguir las especificaciones de hacienda) y el PDF al cliente. Revisar ejemplos anexos.

***NOTA: Se quiere evitar información duplicada para todo lo relacionado con el Usuario por lo tanto el Inicio de Sesión y registrar usuario deben realizarse en el sistema actual de TecAirlines.***

* Creación de BagCart: El sistema debe proveer la creación de un carrito de transporte de maletas del cual se almacenará un identificador, la marca (DropDown de selección y se pueden incluir más marcas) y Modelo (Año).

**NOTA:** Los vuelos y aviones no se crearán con la aplicación pero deben ser populados en el el script de populación al menos 50 vuelos.

* Asignación de un avión a un vuelo: Cada uno de los vuelos tiene un tipo de avión que indica las secciones de bodega que tiene.
* Asignación de un BagCart a un vuelo: Para cada vuelo abierto se debe asignar un BagCart para iniciar el proceso de abordaje de maletas.
* Cierre de un BagCart: Cada vez que se asigne un BagCart a un vuelo cuando este es despachado al avión, se debe poner un sello de seguridad, este es un alfanumérico de 10 caracteres.
* Reportes:
  + Maletas por cliente.
  + Conciliación de Maletas. Número de Vuelo, Tipo de Avión, Capacidad, Total de maletas del vuelos, Cantidad de maletas en BagCarts, total de maletas en el avión, rechazadas.

**Aplicación Web móvil:**

* Inicio de Sesión. Igual que en la parte administrativa.
* Asignar/Escaneo de una Maleta a un BagCart: Este proceso es super delicado ya que a la hora de asignar una maleta a un BagCart el usario que lo realiza certifica que esa maleta paso por Rayos X y no poseía ningún contenido prohibido (armas, droga, etc). En este caso se debe poder revisar quien registro (escaneo en Rayos X) una maleta.
* Rechazo de una maleta. En caso de que los rayos X revelen que existe un artículo no permitido el sistema debe permitir poner la maleta en estado rechazada y habilitar un comentario de porque se rechazo.
* Asignación de maletas a un avión: Este proceso consiste en escanear una maleta y asignarla una bodega del avión, debe quedar registro de quien escaneo una maleta a la hora de asignarla al avión.
  1. **Requerimientos NO funcionales del Sistema**
* Existirán tres vistas.
* Recepcionista y administrador. Scan y Embarcador. A la hora de realizar el login debe seleccionar que rol utilizara.
* El Sistema debe ser una aplicación web (utilizando Angular, Bootstrap, HTML5 y CSS).
* La Base de Datos debe estar en postgreSQL.
* La capa de servicios debe estar desarrollada en C# y debe ser desplegada en la nuble AWS/Azure **NO** **se permite desplegarla en IIS**.
* La Base de Datos debe estar al menos en tercera forma normal.
* Se deben implementar al menos 5 procedimientos almacenados y 2 triggers.

1. **Entregables**
   1. Manual de Usuario.
   2. Documentación Técnica y del proyecto (descrita en el punto 6).
   3. Documento de instalación.
   4. Plan de Proyecto.
   5. Script de Base de Datos.
   6. Script de populación de Base de Datos.
   7. Aplicación WEB.
   8. Web Service/REST Service.
   9. Minutas.
2. **Documentación** 
   1. Se deberá documentar el código fuente.
   2. Se deberá entregar un documento que contenga:
      1. Modelo conceptual utilizando la notación de Chen.
      2. Modelo relacional.
      3. Descripción de las estructuras de datos desarrolladas (Tablas).
      4. Descripción de los Store Procedures/Funciones/triggers implementados.
      5. Descripción detallada de la arquitectura desarrollada.
      6. Problemas conocidos: En esta sección se detalla cualquier problema que no se ha podido solucionar en el trabajo.
      7. Problemas encontrados: descripción detallada, intentos de solución sin éxito, soluciones encontradas con su descripción detallada, recomendaciones, conclusiones y bibliografía consultada para este problema específico.
      8. Documentación de evidencia del trabajo en equipo.
         1. Actividades planeadas y su responsable. (Plan de trabajo)
         2. Minutas de sesiones de trabajo. (Seguimiento al plan de trabajo)
         3. Actividades realizadas por cada estudiante. (Bitácora en digital, donde se describen las actividades realizadas, desde reuniones con el compañero de trabajo, investigaciones, consultas, etc. Se debe describir todo por más insignificante que sea, esto demostrará si ustedes están trabajando en realidad. Este es su diario de trabajo, llevan seguimiento de todo en el tiempo, imaginen que si un compañero los releva en su trabajo, le bastaría con leer sus bitácoras para seguir el trabajo.
         4. Evidencia de uso de un manejador de código (se recomienda Gitgub).
      9. Conclusiones y Recomendaciones del proyecto.
      10. Bibliografía consultada en todo el proyecto
   3. Diagrama de clases y un documento que explique el porqué del diseño.
3. **Evaluación**
4. El proyecto tendrá un valor de un 80% de la nota final, debe estar funcional.
5. La documentación tendrá un valor de un 20% de la nota final, cumplir con los requerimientos especificados en la documentación no significa que se tienen todos los puntos, se evaluará que la documentación sea coherente, acorde al tamaño del proyecto y el trabajo realizado, no escatimen en documentación.
6. Cada grupo recibirá una nota en cada uno de los siguientes apartados Código y Documentación.
7. El profesor no sólo evaluará la funcionalidad del proyecto, esto quiere decir que aunque el proyecto este 100% funcional esto no implica una nota de un 100, ya que se evaluarán aspectos de calidad de código, aplicación del paradigma OOP, uso de herramientas solicitadas, calidad de documentación interna y externa, trabajo en equipo.
8. No se revisarán funcionalidades parciales, ni funcionalidades no integradas.
9. Es responsabilidad de cada miembro del grupo conocer su código, el profesor puede preguntar a cualquier miembro del grupo que le explique alguna funcionalidad/porción de código.
10. De las notas mencionadas en los puntos 1 y 2 se calculará la Nota Final del Proyecto.
11. Las citas de revisión oficiales serán determinadas por el profesor durante las lecciones o mediante algún medio electrónico.
12. Aún cuando el código y la documentación tienen sus notas por separado, se aplican las siguientes restricciones
    1. Si no se entrega documentación, automáticamente se obtiene una nota de 0.
    2. Si no se utiliza un manejador de código se obtiene una nota de 0.
    3. Si no se entrega el punto 4 de la documentación se obtiene una nota de 0.
    4. Si el código y la documentación no se entregan en la fecha indicada se obtiene una nota de 0.
    5. Si el código no compila se obtendrá una nota de 0, por lo cual se recomienda realizar la defensa con un código funcional.
    6. Si el grupo no cuenta con los equipos necesarios para realizar la revisión y no avisó al profesor de esta situación obtendrá una nota de 0.
    7. El código debe ser desarrollado en C#, en caso contrario se obtendrá una nota de 0.
13. Cada grupo tendrá como máximo 30 minutos para exponer su trabajo al profesor y realizar la defensa de éste, es responsabilidad de los estudiantes mostrar todo el trabajo realizado, por lo cual se recomienda tener todo listo antes de ingresar a la defensa.
14. Cada excepción o error que salga durante la ejecución del proyecto y que se considere debió haber sido contemplada durante el desarrollo del proyecto, se castigará con 2 puntos de la nota final del proyecto.
15. Cada grupo es responsable de llevar los equipos requeridos para la revisión.
16. Durante la revisión únicamente podrán participar los miembros del grupo, asistentes, otros profesores y el coordinador del área.
17. Las revisiones se realizan con los estudiantes matriculados en el curso, cualquier persona fuera de estos y los mencionados en el punto 14, no pueden participar en la revisión.
18. Después de enviada la nota final del proyecto el estudiante tendrá un máximo de 3 días hábiles para presentar un reclamo siempre y cuando la funcionalidad esté completa.
19. **Referencias**

**Bootstrap Themes & Templates** (2018-10-04). Recuperado de: <https://wrapbootstrap.com/>

**How to Write Doc Comments for the Javadoc Tool.** (2018-10-04). Recuperado de: http://www.oracle.com/technetwork/articles/java/index-137868.html

**C# Coding Conventions (C# Programming Guide).** (2018-10-04). Recuperado de: <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions>

**Anexos y estructuras (Ministerio de Hacienda)**. (2018-10-26). Recuperado de: [https://www.hacienda.go.cr/ATV/ComprobanteElectronico/frmAnexosyEstructuras.aspx#](https://www.hacienda.go.cr/ATV/ComprobanteElectronico/frmAnexosyEstructuras.aspx)